Solution Methodologies for the Smallest Enclosing Circle Problem

Sheng Xu[1] **Robert M. Freund**[2] **Jie Sun**[3]

Tribute. We would like to dedicate this paper to Elijah Polak. Professor Polak has made substantial contributions to a truly broad spectrum of topics in nonlinear optimization, including optimization for engineering design centering, multi-criteria optimization, optimal control, feasible directions methods, quasi-Newton and Newton methods, non-differential optimization, semi-infinite optimization, conjugate directions methods, gradient projection and reduced gradient methods, and barrier methods, among many other topics. His many and varied contributions to our field are important today and will influence the research in our field well into the future.

Abstract. Given a set of circles $C = \{c_1, ..., c_n\}$ on the Euclidean plane with centers $\{(a_1, b_1), ..., (a_n, b_n)\}$ and radii $\{r_1, ..., r_n\}$, the smallest enclosing circle (of fixed circles) problem is to find the circle of minimum radius that encloses all circles in C. We survey four known approaches for this problem, including a second order cone reformulation, a subgradient approach, a quadratic programming scheme, and a randomized incremental algorithm. For the last algorithm we also give some implementation details. It turns out the quadratic programming scheme outperforms the other three in our computational experiment.

Keywords: Computational geometry, optimization.

Received September 27, 2001, Revised February 3, 2003

1 *Gintic Institute of Manufacturing Technology, Singapore. E-mail: sxu@gintic.gov.sg.* *This author's work was partially supported by Singapore-MIT Alliance.*

2 *Sloan School of Management, Massachusetts Institute of Technology and Singapore-MIT Alliance. E-mail: rfreund@mit.edu.* *His research was partially supported by Singapore-MIT Alliance.*

3 *Department of Decision Sciences, National University of Singapore and Singapore-MIT Alliance. E-mail: jsun@nus.edu.sg.* *His research was partially supported by grants from Singapore-MIT Alliance and grants R314-000-026/028/042-112 of National University of Singapore.*

1 Introduction

Given a set of circles $C = \{c_1, ..., c_n\}$ on the Euclidean plane with centers $\{(a_1, b_1), ..., (a_n, b_n)\}$ and radii $\{r_1, ..., r_n\}$, the smallest enclosing circle (of fixed-center circles) problem is to find the circle of minimum radius that encloses all circles in C.

The smallest enclosing circle problem arises in various application areas, such as:

- To plan the location of a shared facility. For example, to build a hospital servicing a number of circle-shaped communities so as to minimize the distance between the hospital and the farthest community from it.
- To rotate a set of circles into any arbitrary alignment: The minimal enclosing circle of the points is the amount of space on the plane that must be empty to permit this rotation.
- To provide a rough approximation to the points of the minimal enclosing circle: Through the test of proximity between sets of points/circles and the computation of the minimal enclosing circle, the distance between a new point and a given set could be quickly determined.
- To solve problems in environmental science (design and optimization of solvents), pattern recognition (finding reference points), biology (protein analysis), political science (analysis of party spectra), mechanical engineering (stress optimization), and computer graphics (ray tracing, culling) etc. More descriptions of these applications can be found in [2, 5, 7].

Most algorithms discussed in the literature consider only the special case of enclosing points, namely the case of $r_i = 0, i = 1, ..., n$. The simplest algorithm is to check every circle defined by two or three of the n points (we say a circle is defined by two points if its diameter is the line segment between the two points) and finds the smallest of these which contains every point. There are $O(n^3)$ such circles and each takes $O(n)$ time to check, so the total running time is $O(n^4)$. This algorithm is computationally expensive and improvements date back as early as in 1860, see [7]. While the most theoretically efficient method so far is shown to have an $O(n)$ complexity (Megiddo [10]), a lot of practically efficient methods exist for the enclosing point problem, see Chrystal [4], Elliosoff and Unger [5], Elzinga and Hearn [6], Gärtner [7], Hearn and Vijan [8], and Welzl [12]. For the general cases where at least one circle has non-zero radius, the problem can be formulated as a convex program (See Section 2.1 below). Therefore, any suitable methods of convex programming can in principle be used for solving this problem. The key point is which one is more efficient for large

practical problems, which can only be answered by practical testing. In this paper, we survey four approaches that come with very different theoretical background (second-order cone optimization, subdifferential optimization, quadratic optimization, and combinatorial optimization) and report our numerical results in testing randomly generated problems.

2 Four algorithms for the smallest enclosing circle problem

In this section we survey four algorithms for the smallest enclosing circle (of fixed circles) problem. Two of the algorithms have theoretical proofs of convergence while the other two are heuristics. All four algorithms converge on our numerical tests of several thousand problems, demonstrating that all four methods are very robust. The test results of these methods are reported in Section 3.

2.1 A formulation as a second order cone optimization problem

The most direct way to solve the enclosing circle problem is to formulate the problem as a second order cone optimization problem.

Suppose the center of the enclosing circle is (x, y), and its radius is R. The smallest enclosing circle problem can be formulated as a constrained convex program as follows.

$$\min\ R \quad \text{s.t.} \quad \sqrt{(x-a_i)^2+(y-b_i)^2}+r_i \le R \qquad i=1,...,n. \tag{2.1}$$

By introducing auxiliary variables $x_i = a_i - x$, $y_i = b_i - y$, and $z_i = R - r_i$, $i = 1, ..., n$, this problem can be reformulated as follows:

$$\begin{aligned} \min \quad & R \\ \text{s.t.} \quad & x_i + x = a_i \qquad i=1,...,n \\ & y_i + y = b_i \qquad i=1,...,n \\ & z_i + r_i = R \qquad i=1,...,n \\ & z_i \ge \sqrt{x_i^2+y_i^2} \qquad i=1,...,n. \end{aligned}$$

This formulation is a second order cone optimization problem, for which there are interior point algorithms of iteration order $O(\sqrt{n}|\log\epsilon|)$ (e.g., Lobo et al [9]) and related packages (e.g., Sturm [11]), where ϵ is the user specified optimality tolerance. Since each iteration typically utilizes $O(n^3)$ arithmetic operations [3], the complexity of this type of algorithms is $O(n^{3.5}|\log\epsilon|)$.

2.2 A subgradient method

Another formulation of our problem is an unconstrained non-differentiable convex program

$$\min \ f(x,y) = \max_{i=1,\ldots,n} \left\{ \sqrt{(a_i - x)^2 + (b_i - y)^2} + r_i \right\}.$$

The recent paper of Barahona and Anbil [1] shows that the subgradient method can be very effective in solving extremely large linear programs, which motivates the following heuristic approach.

Step 1: Starting point: $x_0 = (a_1 + a_2 + \cdots + a_n)/n$, $y_0 = (b_1 + b_2 + \cdots + b_n)/n$. Set $k := 0$

Step 2: Compute a subgradient of $f(x,y)$ at (x_k, y_k).

Step 3: Use line search to determine the step length.

Step 4: If the step length is greater than 0, then $k := k + 1$, and proceed to Step 3. Otherwise, stop.

In Step 3, to compute the subgradient at (x_k, y_k), the following procedure is used. Let $I = \{i | f(x,y) = \sqrt{(a_i - x)^2 + (b_i - y)^2} + r_i\}$ be the index set of the active functions in function $f(x,y)$. Then it is well known that

$$\partial f(x,y) = \text{co} \left\{ \left(\sqrt{(a_i - x)^2 + (b_i - y)^2} \right)^{-1} \begin{pmatrix} x - a_i \\ y - b_i \end{pmatrix} : i \in I \right\}, \tag{2.2}$$

where $\partial f(x,y)$ is the subdifferential of f at (x,y) and "co" stands for the convex hull. Thus, any element in this set is a subgradient of $f(x,y)$. In particular, if this set is a singleton, then $f(x,y)$ is differentiable at this point and we obtain the steepest descent direction by taking the search direction as $(a_i - x, b_i - y), i \in I$ (the negative gradient direction). If there is more than one index in I, then we take the negative subgradient of smallest norm, namely we compute the search direction as the projection of 0 onto $-\partial f(x,y)$.

We give a geometric interpretation of the algorithm. At (x_k, y_k) we look for the farthest circle from (x_k, y_k), where the distance from (x_k, y_k) to each circle c_i is $d_i = \sqrt{(a_i - x_k)^2 + (b_i - y_k)^2} + r_i$, for $i = 1, ..., n$. If there is only one such circle, we move toward its center to reduce the radius of the enclosing circle, which is the direction $(a_i - x_k, b_i - y_k)^T$. If there more than one farthest circle, say $\{c_i \mid i \in I\}$, we move using the projection of 0 onto $-\partial f(x_k, y_k)$, which is the polytope generated by the unit vectors along the directions $(a_i - x_k, b_i - y_k)^T$, $i \in I$.

2.3 A quadratic programming approach

The smallest enclosing circle can be found by solving a sequence of convex quadratic programming problems. For this purpose, we first introduce a new variable θ and consider for fixed $R \geq \max\{r_i, 1 \leq i \leq n\}$ the problem

$$\min\ \theta \text{ s.t. } (x - a_i)^2 + (y - b_i)^2 - \theta \leq (R - r_i)^2 \qquad i = 1, ..., n. \tag{2.3}$$

Theorem 2.1 *Problem (2.3) is equivalent to problem (2.1) in the sense that (x^*, y^*, R^*) is an optimal solution of (2.1) if and only if $(x^*, y^*, 0)$ is an optimal solution of (2.3) for $R = R^*$.*

Proof: Let (x^*, y^*, R^*) be an optimal solution of (2.1) and solve (2.3) for $R = R^*$. By the feasibility of $(x^*, y^*, 0)$ to (2.3) one has $\theta^* \leq 0$. It is obvious that one can not have $\theta^* < 0$ since if there exists a feasible solution $(\bar{x}, \bar{y}, \bar{\theta})$ of (2.3) such that $\bar{\theta} < 0$, then $(\bar{x}, \bar{y}, R^*)$ is a feasible solution of (2.1) such that all of its constraints are satisfied with strict inequalities. Therefore R^* is not optimal to (2.1), a contradiction. This implies that $\theta^* = 0$ in problem (2.3).

Conversely, if the optimal value θ^* of (2.3) is equal to zero, then R must be equal to the optimal value R^* of (2.1). Otherwise, let $R < R^*$ and let (x, y, θ^*) be a feasible solution of (2.3). Then for $R = R^*$, all inequalities in the constraint system are strict for the same x, y and θ^*, which further implies that the optimal value of θ is negative or $-\infty$. Thus, $\theta = 0$ is not the optimal value of (2.3). The case of $R > R^*$ can be proved similarly. In this case let (x^*, y^*, R^*) be an optimal solution to (2.1). Then it is easy to show that $(x^*, y^*, 0)$ is a feasible solution of (2.3) for $R > R^*$ with all constraints being satisfied with strict inequalities. This again contradicts with $\theta^* = 0$. Finally, it is easy to see that if $R = R^*$ and $\theta^* = 0$, then any optimal (x^*, y^*) of (2.3) is also optimal for (2.1). □

Now, we will reformulate problem (2.3) as a series of linearly constrained quadratic programming problems.

Define $z = x^2 + y^2 - \theta$, this problem can be reformulated as follows:

$$\min\ x^2 + y^2 - z \quad \text{s.t.} \quad -2a_i x - 2b_i y + z \leq (R - r_i)^2 - a_i^2 - b_i^2 \qquad i = 1, ..., n. \tag{2.4}$$

If R is fixed and sufficiently large, then problem (2.4) is a linearly constrained quadratic programming problem, which can be solved efficiently. If its optimal value is zero, then R is optimal for

(2.1) and we are done. Otherwise, we reduce R and solve the new quadratic program. Based on this idea, we propose the following algorithm to solve the smallest enclosing circle problem:

Step 1: Start with $(x, y) = (0, 0)$ and compute $R = \max_i \left\{ \sqrt{(a_i - x)^2 + (b_i - y)^2} + r_i \right\}$.

Step 2: Solve problem (3.3) to find z, x, and y.

Step 3: If $|x^2 + y^2 - z| \leq \varepsilon$ (where ε is a predefined tolerance), then stop, otherwise re-compute $R = \max_i (\sqrt{(a_i - x)^2 + (b_i - y)^2} + r_i)$, and proceed to Step 2.

Remarks. The main idea of the method is to solve a series of quadratic programming problems so that θ is always negative, and the results of (x, y) are used to find a better estimate of R at each iteration.

2.4 A randomized incremental algorithm

The last method we study is a randomized incremental algorithm, which was first proposed by Welzl [12]. Suppose that none of the given circles is contained in another given circle. By using the fact that the smallest enclosing circle should be tangent to one, two, or three of the given circles (see proof below), we could start from a single circle and enlarge it gradually to enclose all given circles.

Definition 2.2 *Circle a is said to lie on the boundary of circle b if circle a is contained by circle b and is an internally tangent circle of circle b.*

Definition 2.3 *Given a set C of n circles in the plane, let $sb(C, B)$ denote the circle of smallest radius containing all circles in C, where B is the subset of C, containing all circles lying on the boundary of $sb(C, B)$. We define $sb(C, B) = \emptyset$ if $C = \emptyset$.*

Clearly, when $n = 1$, one has $C = \{c_1\}$, $sb(C, B) = sb(\{c_1\}, \{c_1\}) = c_1$. When $n = 2$, since none of the circle is contained by the other, one has $sb(C, B) = sb(\{c_1, c_2\}, \{c_1, c_2\})$ and it is easy to construct this circle, namely the circle whose center lies on the line segment between the two centers of c_1 and c_2 and have the two given circles lie on its boundary.

It is also clear that for $n \geq 3$ one has $sb(C, B) = sb(B, B)$ and $B \neq \emptyset$ (These are always true if $C \neq \emptyset$ and will be shown in Corollary 2.5 below). Thus, the smallest enclosing ball is completely determined by B. Note that $sb(B, B)$ is easy to compute: If $|B|$ (the cardinality of B) is one or

two, then $sb(B, B)$ is either $sb(\{c_1\}, \{c_1\})$ or $sb(\{c_1, c_2\}, \{c_1, c_2\})$; if $|B| \geq 3$, then take arbitrary three circles and solve the following equation system.

$$(x - a_1)^2 + (y - b_1)^2 = (R - r_1)^2 \tag{2.5}$$
$$(x - a_2)^2 + (y - b_2)^2 = (R - r_2)^2 \tag{2.6}$$
$$(x - a_3)^2 + (y - b_3)^2 = (R - r_3)^2 \tag{2.7}$$

Subtracting (2.5) − (2.6), (2.5) − (2.7), and (2.7) − (2.6) yields the following linear equations:

$$2(a_2 - a_1)x + 2(b_2 - b_1)y + 2(r_1 - r_2)R = r_1^2 - a_1^2 - b_1^2 - (r_2^2 - a_2^2 - b_2^2) \tag{2.8}$$
$$2(a_3 - a_1)x + 2(b_3 - b_1)y + 2(r_1 - r_3)R = r_1^2 - a_1^2 - b_1^2 - (r_3^2 - a_3^2 - b_3^2) \tag{2.9}$$
$$2(a_2 - a_3)x + 2(b_2 - b_3)y + 2(r_3 - r_2)R = r_3^2 - a_3^2 - b_3^2 - (r_2^2 - a_2^2 - b_2^2) \tag{2.10}$$

From any two of equations (2.8), (2.9), and (2.10), we obtain x, y as functions of R. For example, we use equations (2.8), (2.9) to derive:

$$x + \frac{b_2 - b_1}{a_2 - a_1}y + \frac{r_1 - r_2}{a_2 - a_1}R = \frac{r_1^2 - a_1^2 - b_1^2 - (r_2^2 - a_2^2 - b_2^2)}{2(a_2 - a_1)} \tag{2.11}$$
$$x + \frac{b_3 - b_1}{a_3 - a_1}y + \frac{r_1 - r_3}{a_3 - a_1}R = \frac{r_1^2 - a_1^2 - b_1^2 - (r_3^2 - a_3^2 - b_3^2)}{2(a_3 - a_1)}, \tag{2.12}$$

which gives

$$\begin{aligned} x = \quad & \left(\frac{r_1^2 - a_1^2 - b_1^2 - (r_2^2 - a_2^2 - b_2^2)}{2(b_2 - b_1)} - \frac{r_1^2 - a_1^2 - b_1^2 - (r_3^2 - a_3^2 - b_3^2)}{2(b_3 - b_1)}\right) / \left(\frac{a_2 - a_1}{b_2 - b_1} - \frac{a_3 - a_1}{b_3 - b_1}\right) \\ & -R\left(\frac{r_1 - r_2}{b_2 - b_1} - \frac{r_1 - r_3}{b_3 - b_1}\right) / \left(\frac{a_2 - a_1}{b_2 - b_1} - \frac{a_3 - a_1}{b_3 - b_1}\right) \end{aligned} \tag{2.13}$$

$$\begin{aligned} y = \quad & \left(\frac{r_1^2 - a_1^2 - b_1^2 - (r_2^2 - a_2^2 - b_2^2)}{2(a_2 - a_1)} - \frac{r_1^2 - a_1^2 - b_1^2 - (r_3^2 - a_3^2 - b_3^2)}{2(a_3 - a_1)}\right) / \left(\frac{b_2 - b_1}{a_2 - a_1} - \frac{b_3 - b_1}{a_3 - a_1}\right) \\ & -R\left(\frac{r_1 - r_2}{a_2 - a_1} - \frac{r_1 - r_3}{a_3 - a_1}\right) / \left(\frac{b_2 - b_1}{a_2 - a_1} - \frac{b_3 - b_1}{a_3 - a_1}\right) \end{aligned} \tag{2.14}$$

Substitute (2.13) and (2.14) in any of the three equations (2.5), (2.6), or (2.7) to compute R and then use equations (2.13) and (2.14) to compute x and y. The existence of solution is guaranteed by the non-colinearity of the three centers.

In summary, the determination of the smallest enclosing circle is equivalent to the determination of B, whereas $|B| \geq 3$ we only need three arbitrary circles in B.

Let $C_k = \{c_1, ..., c_k\}$ be a set of $k \leq n$ circles randomly chosen from C and let $sb(C_k, B_k)$ be the smallest enclosing ball of C_k. We note that B_k can be recursively computed from $B_1 = \{c_1\}$. Before we state the algorithm, we need the following proposition.

Proposition 2.4 *If $c_k \not\subset sb(C_{k-1}, B_{k-1})$, then $c_k \in B_k$.*

Proof. Let the center and radius of $sb(C_k, B_k)$ be (x, y) and R, respectively. Then the KKT conditions give

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \sum_{i=1}^{k} \lambda_i \begin{pmatrix} x - a_i \\ y - b_i \\ r_i - R \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{2.15}$$

$$\lambda_i \left[(x - a_i)^2 + (y - b_i)^2 - (R - r_i)^2 \right] = 0, \ i = 1, 2, ..., k, \tag{2.16}$$

$$\lambda_i \geq 0, (x - a_i)^2 + (y - b_i)^2 \leq (R - r_i)^2, \ i = 1, 2, ..., k. \tag{2.17}$$

If $\lambda_k = 0$ the conditions imply that $sb(C_k, B_k) = sb(C_{k-1}, B_{k-1})$. Thus $c_k \subset sb(C_{k-1}, B_{k-1})$, which contradicts with the assumption. Hence one obtains $\lambda_k \neq 0$, which implies $(x - a_k)^2 + (y - b_k)^2 = (R - r_k)^2$ by (2.16). The equation shows that c_k lies on the boundary of $sb(C_k, B_k)$. □

Corollary 2.5 *$sb(C, B) = sb(B, B)$ and if $C \neq \emptyset$ then $B \neq \emptyset$.*

Proof. Note that any (x, y) and R that satisfy (2.15) – (2.17) also satisfy the KKT conditions for $sb(B, B)$ which is the same as (2.15) – (2.17) except deleting the terms corresponding to $(x - a_i)^2 + (y - b_i)^2 < (R - r_i)^2$, i.e., the non-boundary circles. Hence $sb(C, B) = sb(B, B)$.

The proof of the second part is also trivial: Just note that if $B = \emptyset$ then all $\lambda_i = 0, \ i = 1, ..., k$ in (2.15), leading to $1 = 0$ in (2.15), a contradiction. □

The results above help to develop the following recursive algorithm, which we call the randomized incremental algorithm. Its pseudo code is as follows.

Algorithm 2.6 *% The Randomized Incremental Algorithm*

% C is a set of circles to be enclosed, and B is a set of boundary circles

FUNCTION miniCircle(C,B);

IF C: = $\emptyset$ THEN

IF B:=$\emptyset$ THEN

D:=∅;

ELSE IF B:={p} THEN

D:=p;

ELSE IF B:={p, q} THEN

D:= the smallest enclosing circle of p and q;

END IF

ELSE

Choose random $c_i \in C$*;*

D:=miniCircle$(C - \{c_i\}, B)$*;*

IF $c_i \notin D$ *THEN* *%*c_i *is on the boundary of D*

IF the number of elements in set $B \leq 1$ *THEN*

D:= miniCircle$(C - \{c_i\}, B + \{c_i\})$*;*

ELSE *% Three boundary circles are found*

Calculate D's radius R and D's center (x, y) *using equations (2.5), (2.13), and (2.14);*

END IF

END IF;

END

RETURN D;

Justification of the algorithm. We compute the smallest enclosing circle $sb(C, B)$ by recursively computing $sb(C_k, B_k)$ from $sb(C_1, B_1) = \{c_1\}$. We justify the algorithm by induction.

At step $k + 1$, we have $sb(C_k, B_k)$ and randomly select another circle c_{k+1}. If $c_{k+1} \subset sb(C_k, B_k)$, then $B_{k+1} = B_k$ and $sb(C_{k+1}, B_{k+1}) = sb(C_k, B_k)$. Otherwise, we start with $B_{k+1} = \{c_{k+1}\}$ and call to $miniCircle$ which computes the smallest circle enclosing $\{c_1, c_2, ..., c_k\}$ with c_{k+1} on its boundary.

The procedure is as follows. We randomly pick a circle among $C = \{c_1, c_2, ..., c_k\}$, say c_1, and construct the smallest circle enclosing $\{c_2, ..., c_k\}$ with c_{k+1} on its boundary. According to the induction supposition, such a circle D is available. We then ask whether $c_1 \subset D$. If yes, then $D = sb(C_{k+1}, B_{k+1})$ (obvious); if not, then both c_1 and c_{k+1} should be on the boundary of $sb(C_{k+1}, B_{k+1})$ (Proposition 2.4). For convenient of terminology, let us call this process the layer-1 process.

In the "not" case of layer-1, we then pick another circle c_2 randomly and construct the smallest circle enclosing $\{c_3, ..., c_k\}$ with c_1, c_{k+1} on its boundary. According to the induction supposition, such a circle is available and we also call it D to be consistent with the pseudo code. We ask if $c_2 \subset D$. If yes, then $D = sb(C_{k+1}, B_{k+1})$ (obvious); if not, we are done because $B_{k+1} = \{c_{k+1}, c_1, c_2\}$ (Proposition 2.4). Let us call this process the layer-2 process.

Thus, the case of $k+1$ circles is reduced to the case of k or $k-1$ circles. At each step, the algorithm generates $sb(C_k, B_k)$. By induction, the algorithm is justified. □

Note that the algorithm is almost the same as the randomized incremental algorithm of Welzl [12]. The change is that we check whether a *circle*, rather than a *point*, is in the disk D at each iteration. If the answer is "yes", then both algorithms step into the next iteration. If the answer is "no", then both algorithms recursively proceed to the case of $n-1$ circles or points with one more circle or point on the boundary. When constructing the enclosing circle, both algorithms use ≤ 3 boundary circles or points. Thus, the expected complexity of the algorithm here should be the same as Welzl's algorithm.

3 Numerical results

In this section we will compare the numerical performance of all four methods for the smallest enclosing circle of fixed circles problem, namely , the second order cone optimization formulation, the subgradient method, the quadratic programming method, and the randomized incremental algorithm.

We implemented a version of the randomized incremental algorithm, the subgradient method and the quadratic programming method to solve the smallest enclosing circle problem using MATLAB. We also used SeDuMi, a software package for cone optimization problems, to solve the second order cone optimization formulation of the problem. SeDuMi is a MATLAB package with some complied portions, see Sturm [11].

The centers of the circles to be enclosed were generated as normally distributed random points, and radii of these circles were generated as uniformly distributed random numbers and checked so that no circles were contained in any other. We tried problems of different sizes, from 5 circles to 300 circles, and 50 random problems were solved using the four methods for each problem size. Figure 1 shows the average running time of these methods in solving these randomly generated problems where the horizontal axis shows the size n of the problems.

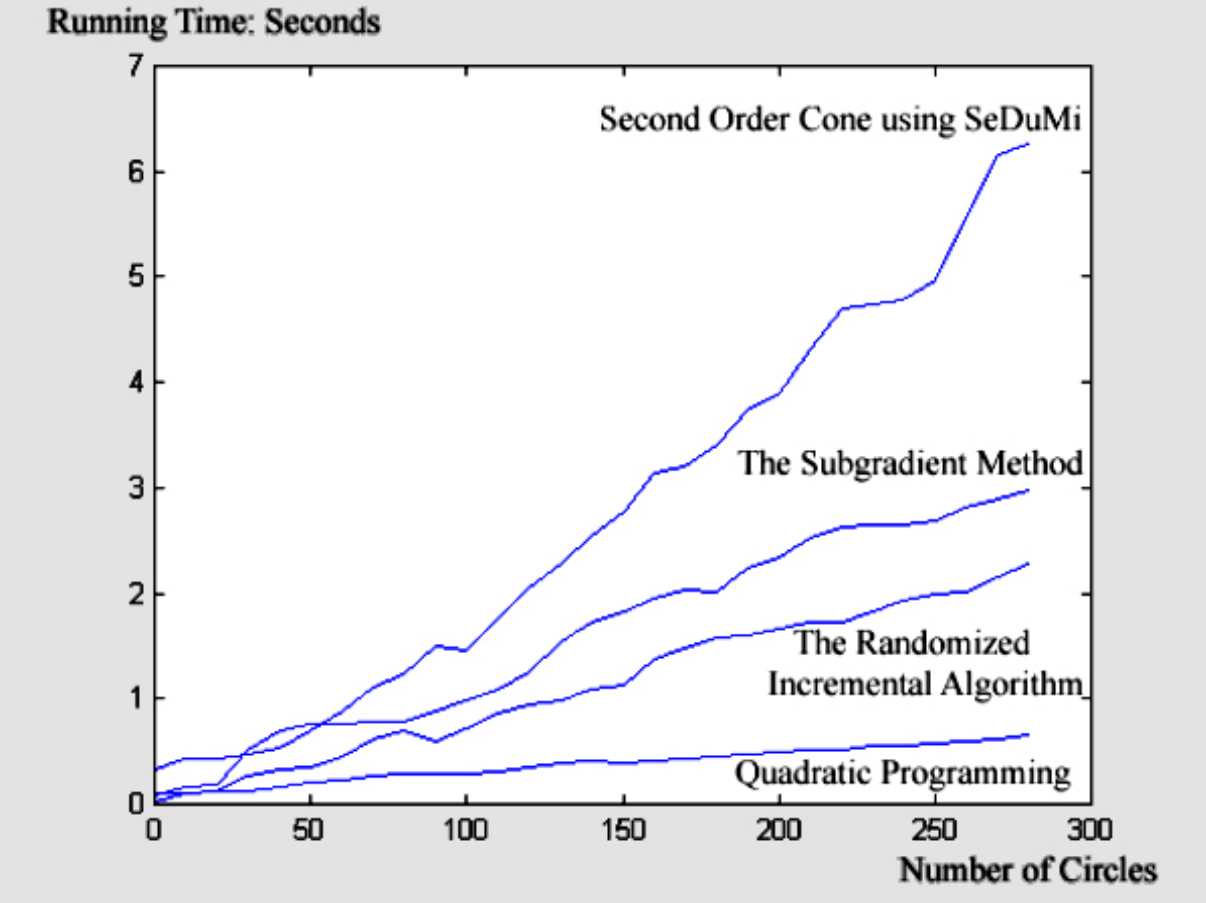

Figure 1: Average running time of four methods

The numerical results show that all methods can find the optimal solution in reasonable time and they perform similarly when the number of circles is small (≤ 50, say). We can also observe that the quadratic programming method performs better than the other three methods when the number of circles increases.

It is observed that the running time of the second order cone formulation is longer on average than that of the other algorithms. It is no surprise the subgradient method is not the best due to its zigzag behavior, a common problem of gradient based methods. To our surprise, however, we note that the "direct" method, namely the randomized incremental algorithm is defeated by the "indirect" quadratic programming method and it appears that as the number of circles increases, the quadratic programming method tends to be more and more efficient than the other three methods.

References

[1] Barahona, F. and P. Anbil (2000), "The volume algorithm: producing primal solutions with a subgradient method", *Mathematical Programming*, **87,** 385-399.

[2] Berg, M. (1997), *Computational Geometry: Algorithms and Applications*, Springer.

[3] Bertsimas, D. and R. Freund (2000), *Data, Models, and Decisions: the Fundamentals of Management Science*, South-Western College Pub.

[4] Chrystal, P. (1885), "On the problem to construct the minimum circle enclosing n given points in a plane", in: *Proceedings of the Edinburgh Mathematical Society*, Third Meeting, p.30.

[5] Eliosoff, J. and R. Unger (1998), "Minimal spanning circle of a set of points", *Computer Science 308-507: Computational Geometry project*, school of computer science, McGill University.

[6] Elzinga, J. and D. Hearn (1972), "The Minimum covering sphere problem", *Management Science.*, **19,** 96-104.

[7] Gärtner, B. (1999), "Fast and robust smallest enclosing balls", Nestril, J., editor, *Algorithms - ESA'99 : 7th Annual European Symposium, proceedings*, Vol. 1643 of *Lecture Notes in Computer Science*, pp. 325- 338, Springer-Verlag.

[8] Hearn, D.W. and J. Vijan (1982), "Efficient algorithms for the minimum circle problem", *Oper. Res.*, **30,** 777-795.

[9] Lobo, M., L. Vandenberghe, S. Boyd, and H. Lebret (1998), "Applications of second-order cone programming", *Linear Algebra and its Applications*, **284,** 193-228.

[10] Megiddo, N. (1983), "Linear-time algorithms for linear programming in $\Re^3$ and related problems", *SIAM J. Comput.*, **12,** 759-776.

[11] Sturm, J. (1999), "Using SeDuMi 1.0x, A MATLAB toolbox for optimization over symmetric cones", http://www.unimaas.nl/ sturm.

[12] Welzl, E. (1991), "Smallest enclosing disks (balls and ellipsoids)", H. Maurer, editor, *New Results and New Trends in Computer Science*, Vol. 555 of *Lecture Notes in Computer Science*, pp. 359-370, Springer-Verlag.